Markku Lepistö

PLC-FUNKTIOKIRJASTON SUUNNITTELU JA TOTEUTUS

Kone- ja tuotantotekniikan koulutusohjelma
Koneautomaation suuntautumisvaihtoehto
2014

samk
Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

PLC-FUNKTIOKIRJASTON SUUNNITTELU JA TOTEUTUS

Lepistö, Markku
Satakunnan ammattikorkeakoulu
Kone- ja tuotantotekniikan koulutusohjelma
Maaliskuu 2014
Ohjaaja: Suvela, Timo
Sivumäärä: 21
Liitteitä:

Asiasanat: PLC, funktio, kirjasto

Tämän opinnäytetyön aiheena oli laatia järjestelmä, jolla hallitaan PLC-ohjelmissa käytettyjen funktiolohkojen kirjastoa ja dokumentaatiota. Keskitetyn tallentamisen etuna on kaikkien ohjelmoijien mahdollisuus päästä selaamaan viimeisimpiä versioita, ilman että joutuisi kyselemään olisiko jollakulla tilanteeseen sopiva lohko. Opinnäytetyössä selvitettiin, mitä vaatimuksia kirjastoon lisättävälle funktiolle on asetettava, jotta funktio soveltuu kirjastoon, mitä sen dokumentaatiosta täytyy selvitä, että sen toiminta tulee selkeästi esille, sekä laadittiin yleiset ohjeet funktioiden ohjelmointiin.

PLC-FUNCTIONLIBRARY DESING AND IMPLEMENTATION

Lepistö, Markku
Satakunnan ammattikorkeakoulu, Satakunta University of Applied Sciences
Degree Programme in Mechanical Engineering
March 2014
Supervisor: Suvela, Timo
Number of pages: 21
Appendices:

Keywords: PLC, function, library

The purpose of this thesis was to create a system for management of a library for PLC-program functions and manuals. Advantage of centralized location for storage is to allow access for all programmers to browse latest versions, without the need to go asking around if someone had made function for current case. In this thesis was sorted out the requirements for a function to be included into library so that it would be compatible, what it’s documentation must contain so that it’s purpose is clear and was laid down general guidelines for function programming.

SISÄLLYS

1 JOHDANTO

Työn tarkoituksena on tehdä JM-Control nimiselle yritykselle PLC-ohjelmien funktioiden ja niiden dokumentaation hallintaan soveltuva järjestelmä, josta olisi helppo etsiä tarpeisiin mahdollisesti soveltuvia komponentteja.

JM-Control on Ulvilassa toimiva automaatio-ohjauksiin erikoistunut yritys, joka suunnittelee ja valmistaa laite-, prosessi- ja linjaohjauksia. Yritys on kasvanut nopeasti viimevuosien aikana ja ohjelmoijien määrän kasvaessa koodin uudelleen käytettävyys on kärsinyt, koska jokaisella on oma tyylinsä tehdä ohjelmaa.

Koska keskitettyä funktiokirjastoa ei ole käytössä, on jokainen suunnitellut erilaisia funktioita joita käyttää projekteissaan. Tämä tarkoittaa, että samaan tehtävään soveltuvia funktioita on useita, mutta käyttötapa ja erityisominaisuudet vaihtelevat. Lisäksi ongelmia muodostaa puutteellinen dokumentaatio. Koska harvasta funktiosta on selkeää tietoa muilla kuin sen laatijalla, tämä aiheuttaa ongelmia valittaessa käytettävää funktiota jos itseltä ei sopivaa löydy. Lisäksi vanhojen ohjelmien muutostöissä hukkaantuu aikaa, jos muutoksen tekijä on eri kuin alkuperäinen ohjelmoija, kun funktion toimintaa joutuu selvittämään koodia tutkimalla.

Keskitetty kirjasto ratkaisee nämä ongelmat tarjoamalla ohjeet joiden mukaan funktiot suunnitellaan ja dokumentoidaan. Näin ohjelmoijat pystyvät tuottamaan funktioita, joiden kierrätettävyys paranee. Lisäksi kirjaston karttuessa on helpompi löytää valmis funktio tarvittavaan tehtävään kuin alkaa suunnitella omaa. Myös myöhemmin tapahtuvat muutokset helpottuvat, kun itselle tuntemattoman funktion toiminnan voi tarkistaa kirjaston dokumenteista.

2 TILAAJAN VAATIMUKSET

Funktioiden tallentaminen kirjastoksi tuo selviä etuja ohjelmoijille nopeuttamalla ohjelmointityöhön kuluvaa kokonaisaikaa. Tämä saavutetaan kun peruskomponentit voidaan hakea suoraan kirjastosta sen sijaan, että joutuisi itse tekemään niitä.

Alussa kirjastolle määriteltiin seuraavia vaatimuksia:

- Sisältää funktion, sen vaatimat oheistiedostot sekä dokumentaation.

  Perusvaatimukset, jotka määrittelevät, mitä kirjastoon on pystyttävä tallentamaan.
- Dokumentaation sisällettävä tiedot parametreistä sekä käytöstä.

  Dokumentaatiosta on selvittävä miten funktiota käytetään ja millaisilla parametreillä sitä kutsutaan.
- Mahdollisuus dokumentoinnin tulostukseen.

  Dokumentaatio on tarvittaessa saatava paperimuotoiseksi helposti, joten siisti tulostusasu on huomioitava.
- Käytettävyys helppoa ylläpidon ja käytön kannalta.

  Jos käytettävyys on hankala, ei kukaan viitsi nähdä vaivaa sen käyttämiseen.
- Sisältää ohjeet kirjastofunktion luomiselle.

  Helpottaa uusien funktioiden luontia, kun on olemassa ohjeet miten tulisi pyrkiä toimimaan.
- Sovelluttava useampien ohjelmistoympäristöjen käyttöön.

  Käytössä useita PLC-ympäristöjä, jotka kaikki täytyisi saadaa samaan järjestelmään.

3 KOODIN UUDELLEENKÄYTTÖ JA KIRJASTOT

3.1 Uudelleenkäytöstä

Ohjelmointityössä on aina pyritty koodin uudelleenkäyttöön uutta ohjelmaa rakennettaessa. Tällä vähennetään ohjelmointiin ja testaukseen kuluvaa aikaa, kun tiedetään, että käytetty koodi on testattua ja toimivaa.

Yksinkertaisin tapa toteuttaa uudelleenkäyttö on suoraan kopioida valmista koodia ohjelmasta tai ohjelmakohdasta toiseen. Tässä tapauksessa ongelmaksi muodostuu helposti koodin pituuden kasvaminen ja sekavoituminen sekä muutosten hallinnan hankaloituminen.

Helpoiten uudelleen käytettävyyden mahdollisuudet huomaa koneautomaatiossa, jossa toimilaitteet ovat samoja. Moottorit, pneumatiikka- ja hydrauliikkasylinterit sekä taajuusmuuttajakäytöt ovat esimerkkejä peruskomponenteista, joita koneista löytyy. Kun toimilaitteen perusohjaus on valmiina komponenttina, voi ohjelmoija keskittyä miettimään, miten toimilaitetta tulee ohjata koneen kokonaistoiminnan kannalta, eikä käyttää aikaa jokaisen eri toimilaitteen miettimiseen.

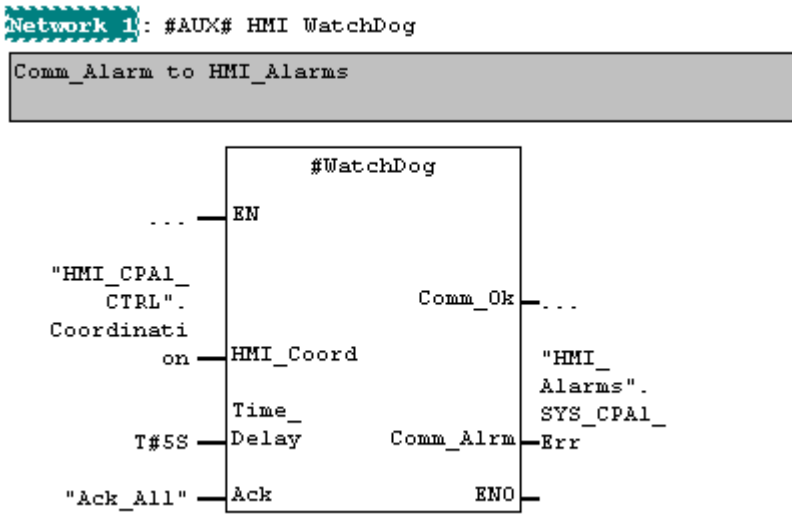
Lisäksi valmiit komponentit helpottavat uusien tai erikoistapausten luontia, kun valmis esimerkki on saatavilla. Esimerkiksi eri valmistajien taajuusmuuttajat toimivat hyvinkin eri tavoilla laitetasolla tarkastellen, mutta ohjauslogiikka ylemmän tason ohjelmasta tarkastellen on kuitenkin sama. Jos on olemassa valmis taajuusmuuttajakomponentti, on sen pohjalta helppo lähteä toteuttamaan uutta versiota, kun ylemmän tason rajapinta on jo valmiiksi mietitty.

3.2 Funktioista

Funktio on itsenäinen aliohjelmalohko, joka suorittaa tietyn tehtävän, kun sitä kutsutaan pääohjelmasta. Funktioita käyttämällä vältetään koodin toistoa ohjelman sisällä ja selkeytetään sitä.

PLC-ohjelmoinnissa laaditaan usein funktio ohjaamaan tietyntyyppistä toimilaitetta tai laitekokonaisuutta. Tällöin funktiolle tuodaan ulkoa tieto laitteen anturoinnilta sekä muut ohjauspyynnöt ja funktio palauttaa uudet ohjausarvot laitteelle. Näin laitteen ohjauksen sisäinen logiikka kätketään funktion sisään ja käyttäjän tarvitsee huolehtia vain funktion kutsumisesta oikeilla parametreillä.

Jotta ohjelmoija pystyy helposti käyttämään funktiota, hänen täytyy tietää, mitä eri parametrit tarkoittavat ja miten ne vaikuttavat funktion toimintaan. Tämän takia funktion dokumentaation täytyy olla mahdollisimman selkeä ja kattava, jotta sen toiminta selviää helposti.

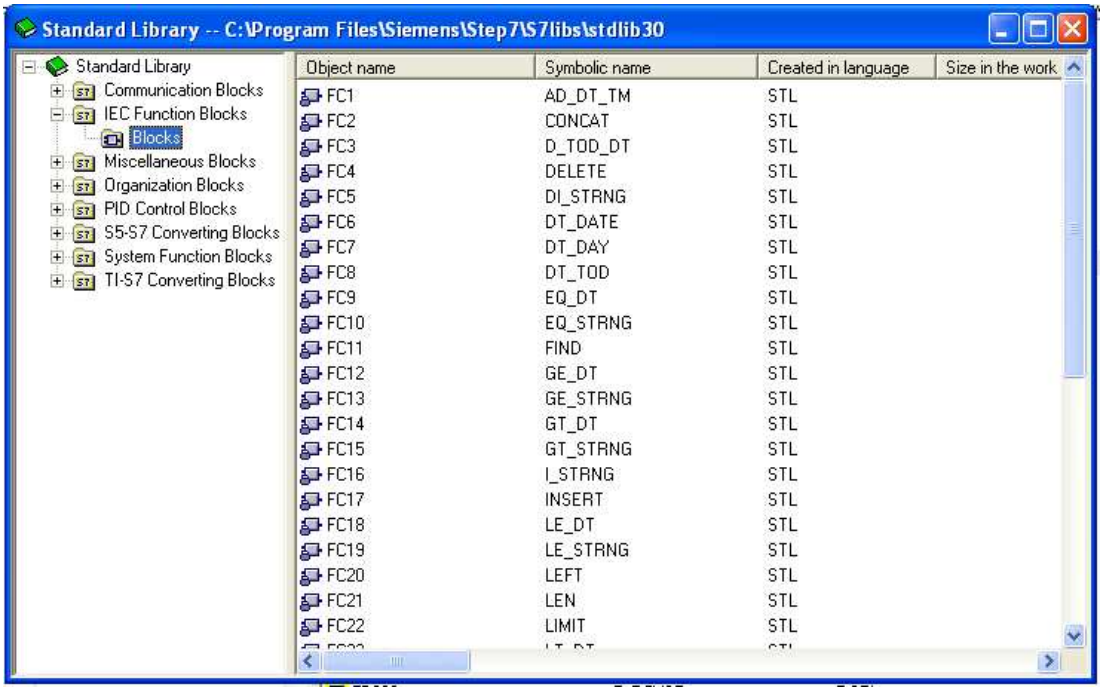


Kuva 1 Funktiokutsu Siemens S7 FBD-kielellä

3.3 Funktiokirjastot

Miltei kaikkiin käytettyihin ohjelmointikieliin on koottu funktiokirjastoja helpottamaan ohjelmoijien työtä. Kirjastot pitävät sisällään kokoelman funktioita sekä mahdollisia datatyyppien määritelmiä ja luokkia. PLC-valmistajat tarjoavat laitteilleen tarkoitettuja kirjastoja tavallisimpiin toimenpiteisiin, kuten matemaattisiin toimintoihin, kommunikointiin jne.

Rakentaessaan ohjelmaa ohjelmoija valitsee kirjastosta tarvitsemansa funktion ja liittää sen osaksi ohjelmaansa, minkä jälkeen hän voi kutsua sitä tarvitsemallaan tavalla.



Kuva 2 Siemens S7 Standard-kirjasto

4 TALLENNUSTAPOJA

4.1 Relaatiotietokannoista

Relaatiotietokantamallin ja käsittelyteorian kehitti Edgar F. Godd vuonna 1970 IBM:lle. Vuosina 1973-1976 IBM kehitti relaatiokannan prototyyppiä nimeltä System R ja kaupallinen versio julkaistiin 1982 nimellä DB2. Nykyisin tarjolla on monien valmistajien tietokantasovelluksia ohjelmaan liitettävistä kirjastoista palvelinohjelmiin.

Relaatiotietokannan perustana on taulu johon tietoa tallennetaan. Taulun sarakkeet ovat tiedon attribuutteja ja rivit tiedon ilmentymiä. Yleensä tietokannassa on useita tauluja jotka linkitetään toisiinsa tiettyjen attribuuttien perusteella. Relaatiomalli määrittää, ettei taulun riveillä ole mitään määrättyä järjestystä, kuten myöskään rivin attribuuteilla ei ole.

Sovellukset hakevat dataa tietokannasta muodostamalla kyselyitä, jotka yleensä kirjoitetaan käyttäen SQL-kieltä (Structured Query Language). Kyselyssä määritetään miten tieto suodatetaan jotta vain halutut rivit saadaan esille. Myös useamman taulun sisältämä tieto voidaan yhdistää join-operaatiolla, jolloin tietokannalle kerrotaan miten taulut liitetään toisiinsa operaatiota varten. Tällä tavoin voidaan muodostaa hyvinkin monimutkaisia hakuja, tiedon esittämiseen.

Tunnetuimpia tietokantajärjestelmiä ovat Oracle, IBM DB2, Microsoft SQL Server, PostgreSQL, MySQL ja SQLite. Osa näistä on vain kaupallisia esim. Oracle ja IBM DB2, kun taas muut käyttävät erilaisia avoimempia lisenssejä tai kaksoislisensointia, joka mahdollistavat ohjelmistojen käytön ilmaiseksi.
[Viitattu 17.1.2012] http://en.wikipedia.org/wiki/Relational_database

4.2 MySQL

MySQL on kirjoitettu käyttäen C ja C++ kieliä, ja se on saatavilla monille käyttöjärjestelmille. Lisäksi useimmille muille ohjelmointikielille on saatavilla tarvittavat aju-

rit yhteyksiä varten. MySQL:n kehitys aloitettiin 1994 kevyempänä vaihtoehtona tehokkaille kaupallisille järjestelmille, mutta se on tasaisesti kehittynyt tukemaan suuremman skaalan tarpeita.

Web-pohjaisissa sovelluksissa MySQL on suosituimpia tietokantoja ja myös suuret ja suositut sivustot kuten Google, Wikipedia ja Facebook käyttävät sitä. GNU General Public License tarjoaa ohjelmiston ja sen lähdekoodin vapaaseen käyttöön, jolloin näitä hyödyntävien ohjelmien ei tarvitse maksaa korkeita lisenssimaksuja.
[Viitattu 17.1.2012] http://en.wikipedia.org/wiki/MySQL

5 KÄYTTÖLIITTYMÄ

Mietittäessä käyttöliittymää tietokannalle oli otettava huomioon usean ihmisen yhtäaikainen käyttö, yksinkertaisuus, dokumentaation tulostaminen tarvittaessa asiakkaille sekä mahdollisuudet kehittää järjestelmää jatkossa. Näitä asioita mietittäessä päädyttiin web-pohjaiseen käyttöliittymään, joka mahdollistaa kirjaston selaamisen myös Internetin kautta ohjelmoijan ollessa asiakkaan luona.

5.1 Web-sivujen ohjelmointi

Tavallisesti web-sivut luodaan käyttäen HTML (Hyper Text Markup Language)-koodausta sekä CSS (Cascading Style Sheets)-tyylimäärityksiä. HTML:n avulla määritellään sivun rakenne ja sisältö esim. otsikot, teksti ja kuvat. Se ei kuitenkaan määrittele, miten sivu kuuluu esittää, vain mitä sivu sisältää. Sivun ulkoasun määrittelyt tehdään CSS:n avulla, jolla kerrotaan, miten sivun elementit kuuluisi esittää.

```
1  <!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transit
2  <html><head><title>Function manual</title>
3      <link rel="stylesheet" href="styles.css" type="t
4      </head><body>
5      <div id="content">
6          <h1>Function creation manual</h1>
7          <h2>General guidelines</h2>
8          <p>Variables should be named as defined in <
9          <p>No references to external variables insid
10             as function IO.</p>
11         <p>Proper and clear commenting so that other
12
13     </div>
14 </body></html>
15
```

Kuva 3. HTML-koodia

Pelkkää HTML:ää ja CSS:ää käyttäen saadaan luotua vain staattisia sivuja, jotka vain esittävät tietoa. Kun halutaan lisätä sivuille toiminnallisuutta, on yleisin vaihtoehto JavaScriptin käyttäminen. JavaScript on web-sivulle upotettava ohjelmointikieli, jonka selain kääntää ja ajaa. Sillä voidaan vaikuttaa sivun rakenteeseen muokkaamalla ja lisäämällä HTML-elementtejä dynaamisesti sekä reagoida käyttäjän syötteeseen. Vaikka JavaScript lisää toiminnallisuutta web-sivuille, ei sillä voida tai kannata kuitenkaan tehdä kaikkea.

Web-sovellusten olennainen osa ovatkin CGI-ohjelmat, jotka ovat web-palvelimella ajettavia ohjelmia. CGI-ohjelmat ovat URL-osoitteita, jotka on määritetty ajettaviksi ohjelmiksi, kun selain kutsuu näitä. Selaimen kutsuessa CGI-ohjelmaa, se generoi web-sivun ensin palvelimella ja lähettää tämän jälkeen valmiin sivun selaimelle. Lisäksi selain voi lähettää tietoa ohjelman käsiteltäväksi, yleisimmin käyttäjän syötettä HTML:n form-elementeiltä. Yksi suosituimmista CGI-kielistä on PHP.

5.2 PHP

PHP:n ensimmäinen versio julkaistiin 1995 nimellä PHP/FI (Personal Home Page / Forms Interpreter) GPL-lisenssin alla. Vuonna 1997 julkaistulla 2.0-versiolla oli tu-

hansia käyttäjiä ja se oli asennettu n. 1 %:iin Internetin domaineista. Vuonna 1998 julkaistiin lähes kokonaan uudelleen kirjoitettu PHP 3 ja viimeisin versio 5.2.4 vuonna 2007.

Nykyään PHP on ehkä suosituin dynaamisten sivustojen laadintaan käytetty kieli. Se toimii sekä Linux että Windows järjestelmissä, sisältää valmiit kirjastot web-formien käsittelyyn sekä useimpien tietokantojen käyttöön. Lisäksi käyttäjillä on mahdollista laajentaa PHP:n toimintaa C:llä kirjoitetuilla lisäosilla.

Yleisimmin PHP:ta käytetään upotettuna html-sivuille. Selaimen pyytäessä sivua, palvelin ajaa koodin PHP-tulkille, joka suorittaa koodin ja palauttaa valmiin sivun. Ohjelma on siis lähdekoodimuodossa palvelimella ja käännetään vasta suoritushetkellä. Tästä aiheutuu pientä tehokkuuden menetystä ylimääräisen käännöstyön takia, mutta ohjelmien muutos on helppoa, kun muutoksen jälkeen ohjelma on heti valmis käyttöön.
[Viitattu 12.9.2008] http://en.wikipedia.org/wiki/Php

```
9         </div>
10    <?php
11    include "settings.php";
12    $id=$_GET['id'];
13
14    // Connect to database
15    $con=mysql_connect($DB_SERVER, $DB_USER, $DB_PASSWD)
16        or die('Could not connect: ' . mysql_error());
17
18    mysql_select_db($DB_NAME) or die('Could not select database');
19
20    $query = "SELECT func_ID, func_info.name, CONCAT(version_major,'.
21        manual, version_minor, short_desc FROM func_info JOIN system
22    $result_info = mysql_query($query) or die('Query failed: ' . mysc
23
24    $row_info = mysql_fetch_array($result_info);
25
26    echo "<h1>$row_info[1]</h1>";
27    echo "<table id=\"meta_info\">";
28    echo "<tr><td>Version:</td><td>$row_info[2]</td></tr>";
29    echo "<tr><td>System:</td><td>$row_info[3]</td></tr>";
30    echo "<tr><td>Author:</td><td>$row_info[4]</td></tr>";
31    echo "<tr><td>Date:</td><td>$row_info[5]</td></tr>";
32    echo "</table> <a class=\"navi\" href=\"download.php?id=$id&versi
```

Kuva 4. PHP-koodia

6 KIRJASTON TOTEUTUS

Yksinkertaisin tapa tallentaa funktiot olisi käyttää verkkoasemaa, jonne funktiot ja dokumentaatio tallennettaisiin erillisinä tiedostoina. Tällöin kirjaston kasvaessa sen hallinta muuttuisi hyvin pian hankalaksi ja etsiminen vaivalloiseksi. Lisäksi pelkän kansiorakenteen käyttö ei mahdollista monimutkaista jäsentelyä eri ryhmiin.

Tiedon jäsentelyn ja hallinnan kannalta paremmaksi vaihtoehdoksi todettiin tietokannan käyttö, jossa dokumentaatio ja tiedosto voidaan linkittää toisiinsa. Lisäksi tietokanta mahdollistaa erilaisen metatiedon lisäämisen funktioiden tietoihin.

6.1 Esimerkkifunktioiden tutkinta

Ensimmäiseksi valittiin joukko funktioita, jotka olisivat käytettävyytensä kannalta soveltuvia kirjastoon. Näiden funktioiden avulla ryhdyttiin määrittelemään tarkemmin mitä dokumentaation tulee sisältää ja mitä vaatimuksia uudelleen käytettävyys esittää itse funktiolle.

Tutkittaessa esimerkkifunktioita selvisi, että parametrien yhtenäisillä nimeämissäännöillä voidaan helposti antaa tietoa, mitä parametri tarkoittaa. Nimeämistä varten päätettiin ryhtyä laatimaan suuntaa antavia ohjeita, joissa listataan yleisesti käytössä olevat lyhenteet parametrinimissä.

Samalla nimeämiskäytäntöä päätettiin laajentaa koskemaan myös ohjelmien symbolitauluja, näin ohjelmaa tehtäessä voi funktioparametrista osittain päätellä mikä symbolinen muuttuja on sille annettava. Tällä saadaan myös helpotettua ohjelman tutkimista jälkikäteen huolto tms. tilanteissa.

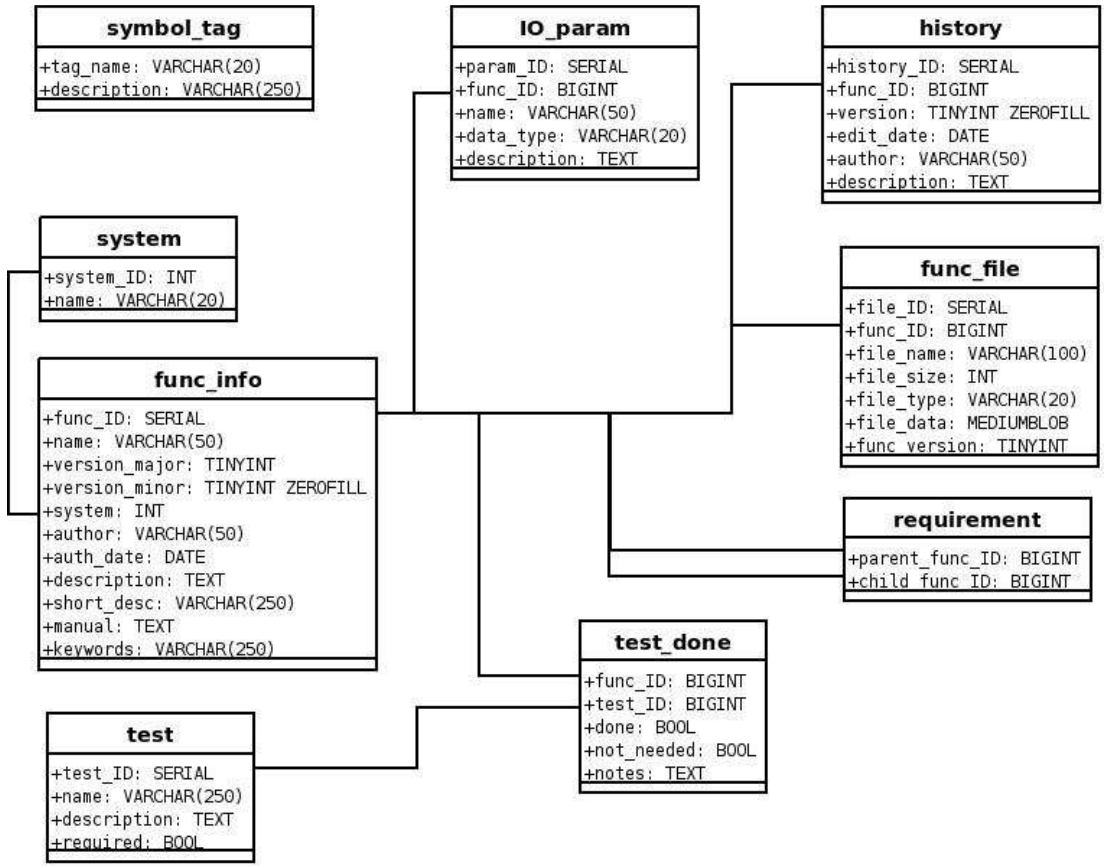
Toinen asia mikä todettiin oli, että funktion sisällä ei saa käyttää mitään ulkoista muuttujaa suoraan, vaan kaikki on tuotava parametreina. Näin vältytään vahingossa tapahtuvalta päällekkäiseltä muistialueen käytöltä, joka voi sekoittaa ohjelman. Hyvänä esimerkkinä Siemens S7 -ympäristössä tarvittavat ”On”- ja ”Off”-bitit (True ja

False), joita ei valmiina ole. Ohjelmoija yleensä siis luo nämä bitit globaalille merkkerialueelle esim. M0.0 ja M0.1. Tämä on kuitenkin ohjelmoijan päätös kyseiselle ohjelmalle, eikä mikään takaa, että toinen ohjelmoija käyttäisi samoja merkkereitä omissa ohjelmissaan. Näin ollen jos funktion sisällä viitattaisiin suoraan globaaliin muuttujaan, esim. ”On” osoitteessa M0.0, ei funktio toimi oikein ohjelmassa jossa M0.0 onkin määritetty olemaan ”Off”.

Funktioiden päivittämistä ja korjauksia varten suunniteltiin säännöt versiohallinnalle. Versionumero jaettiin kahteen osaan, X.Y, joista ensimmäinen kertoo funktion pääversion ja toinen version revision. Revisioiden tarkoituksena on ilmaista funktion sisäisiä muutoksia jotka eivät vaikuta sen ulkoiseen toimintaan. Näin ollen vanhemman revision funktion voi korvata ohjelmassa uudemmalla vaikuttamatta ohjelman toimintaan. Pääversionumero taas päivittyy aina kun funktion toiminta tai ulkoinen liityntä muuttuu niin paljon, että sillä ei voi korvata vanhempaa versiota.

6.2 Toteutus

Toteutus aloitettiin suunnittelemalla dokumentaatiosivun ulkoasu ja kaikki sillä esitettävä tieto. Tämän jälkeen tarvittavaa tietoa ryhdyttiin pilkkomaan osiin joista tietokanta muodostetaan. Tietokannan suunnittelu suoritettiin graafisilla apuvälineillä, joilla saatiin kuvattua taulujen sisältämät sarakkeet ja taulujen väliset yhteydet.
Taulujen suunnittelussa lähtökohtana oli dokumentaation vaatima tieto sisältäen funktion kuvauksen ja parametrit. Tämän jälkeen tauluja lisättiin teknisen toteutuksen vaatimuksien mukaan.



Kuva 5. Tietokannan rakenne

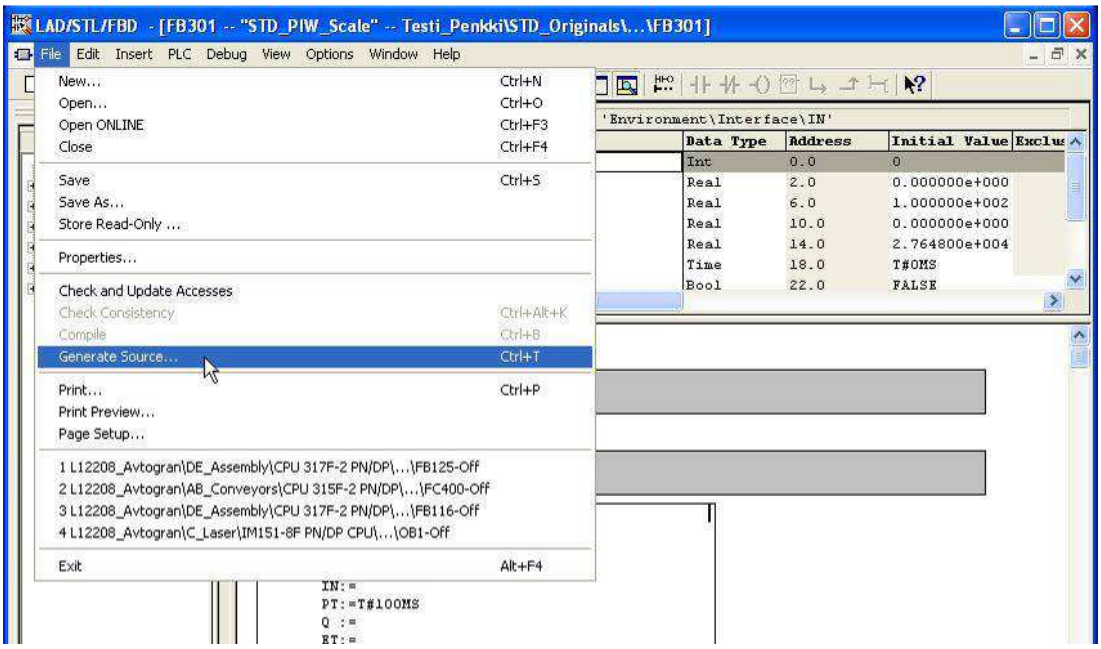
Tarvittava toiminnallisuus mietittiin valmiiksi ja suunniteltiin sivujen ulkoasu, jotka PHP-ohjelma tuottaisi. Itse PHP-ohjelmointi oli varsin suoraviivaista, koska generoitavat web-sivut suunniteltiin yksinkertaisiksi sisältäen vain tarvittavan toiminnallisuuden.

7 KIRJASTON KÄYTTÖ

Funktion tallentamiseksi kirjastoon on se ensin tallennettava muotoon jossa sitä on helppo siirrellä projektista toiseen. Tähän tarkoitukseen PLC-ohjelmointiympäristöt tarjoavat erilaisia tapoja. Ohessa esimerkki Siemens Step7 -ympäristössä.

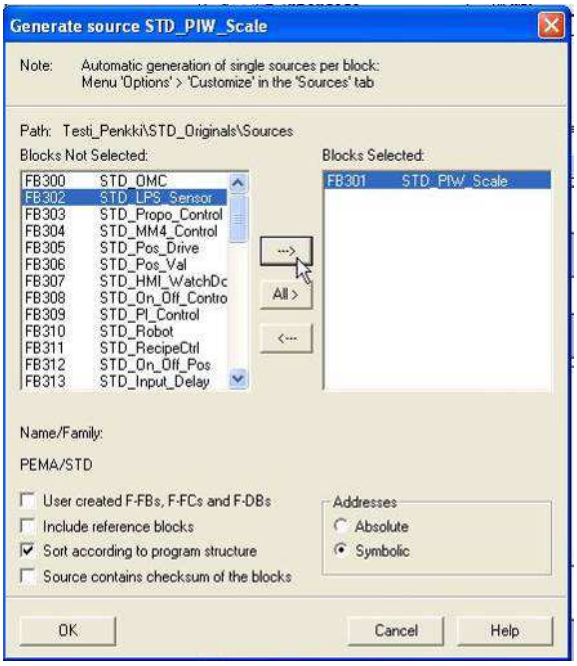
7.1 Funktion vienti ulos ohjelmointiympäristöstä

Ohjelmakomponenttien kopiointi toimii suoraan vain projektista toiseen Step7 -ympäristön sisällä, mutta jotta funktio voidaan irrottaa projektista on siitä ensin tehtävä lähdekoodiobjekti. Tämä tapahtuu ohjelmaeditorin ”Generate Source” -toiminnolla.



Kuva 6. Step7-ohjelma

Valittaessa ”Generate Source” ohjelma kysyy, millä nimellä objekti tallennetaan PLC-ohjelman Sources-hakemistoon. Tämän jälkeen valitaan, mitkä objektit halutaan sisällyttää tiedostoon.

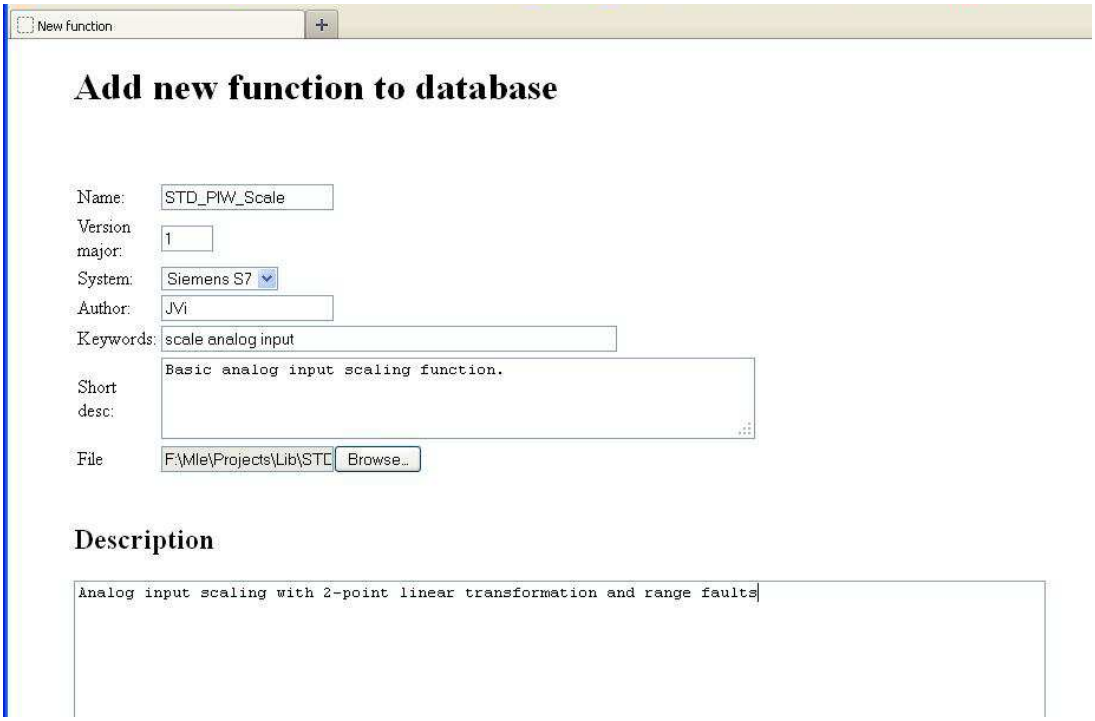


Kuva 7. Funktion valinta

Kun lähdekooditiedosto on luoto, pitää se vielä viedä ulos Simatic Managerista. Tämä tapahtuu valitsemalla haluttu tiedosto Sources-kansiosta ja käyttämällä ”Export Source” -toimintoa. Vasta nyt voidaan tiedosto tallentaa käyttäjän tietokoneella haluttuun kansioon.

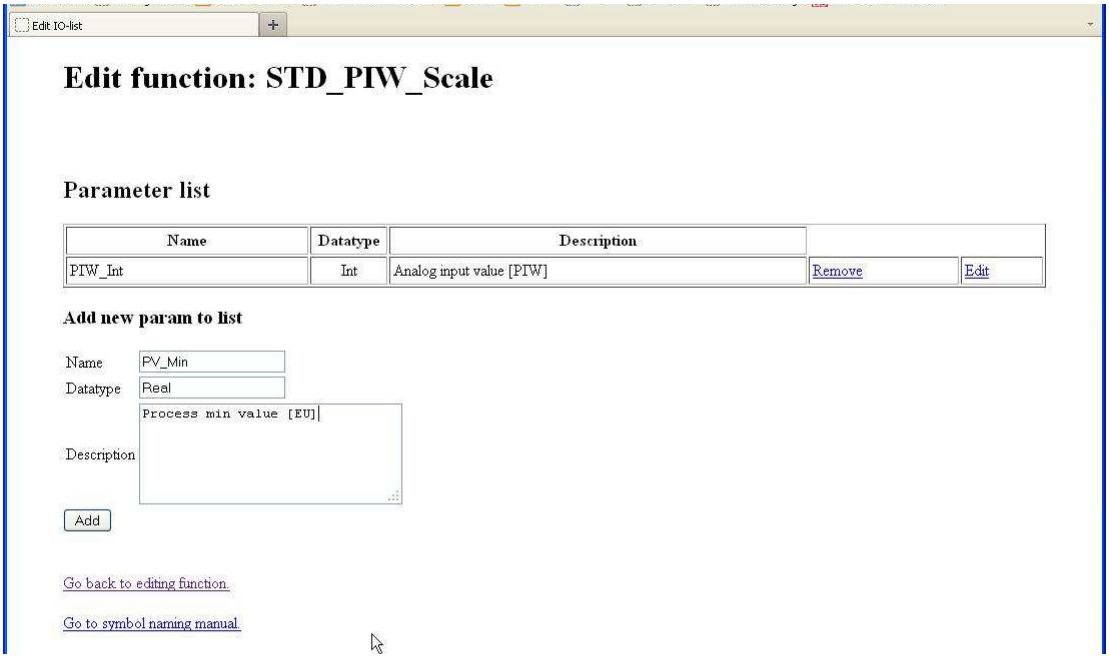
7.2 Lähdekoodin siirto kirjastoon

Lähdekoodi siirretään kirjastoon käyttäen Web-käyttöliittymää. Ensimmäisenä annetaan lisättävän funktion perustiedot sekä itse tiedosto.



Kuva 8. Uuden funktion lisäys

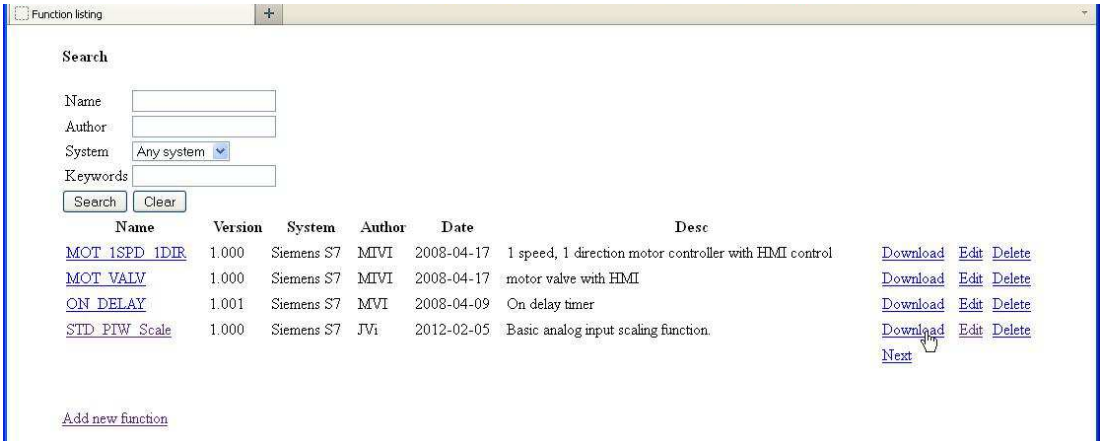
Kun perustiedoilla varustettu funktio on lisätty kirjastoon, voidaan sille antaa lisätietoja, kuten mitä jo kirjastossa olevia funktioita tämän toiminta vaatii, sekä funktion parametrit ja niiden selitykset.



Kuva 9. Parametrien tiedot

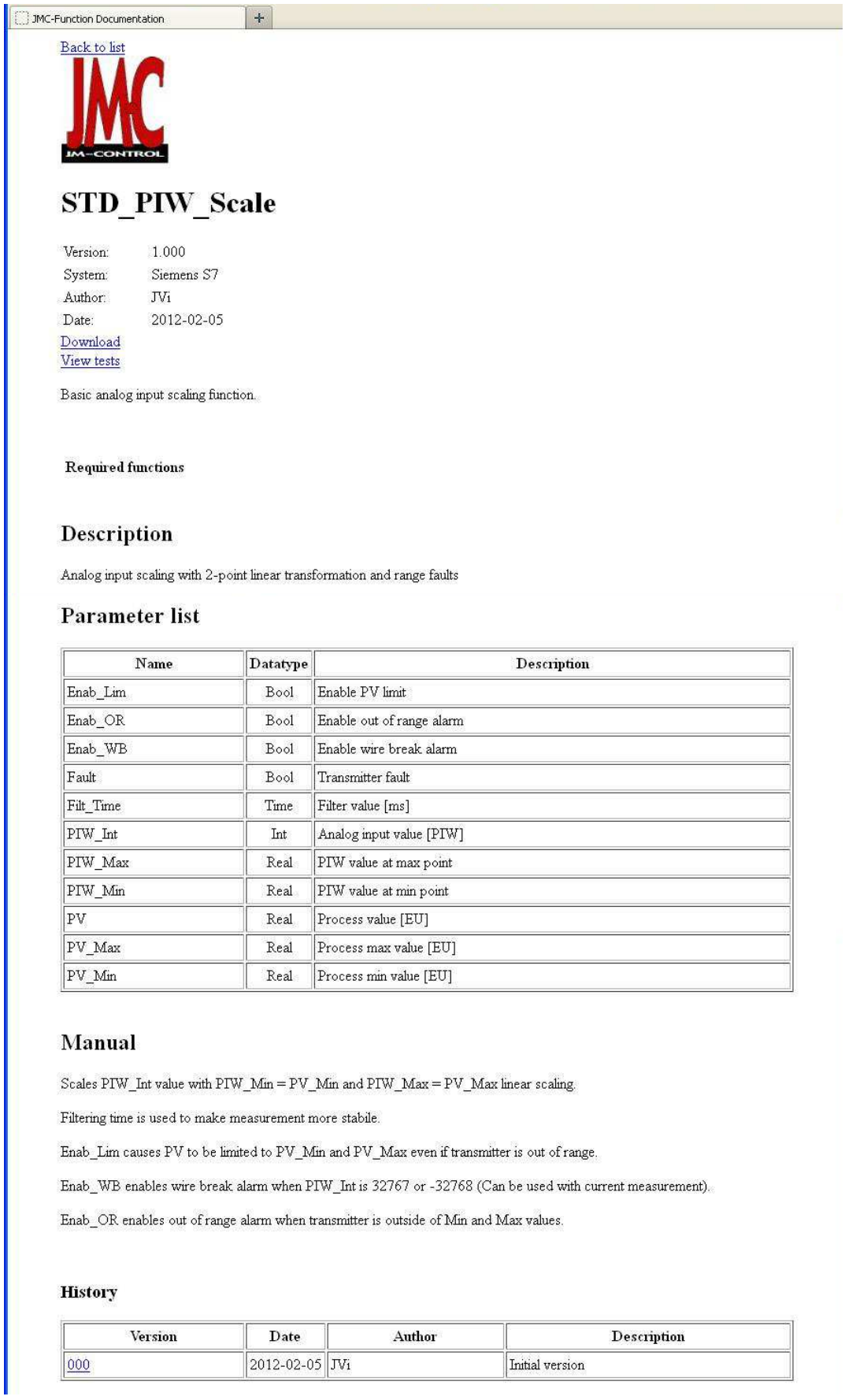
7.3 Funktion haku kirjastosta

Kun funktio on lisätty, se voidaan etsiä kirjastosta yksinkertaisen hakutoiminnon avulla käyttäen joko nimeä, tekijää tai funktiolle annettuja avainsanoja. Lisäksi hakua voidaan rajata, mille ohjelmointiympäristölle laadittuja funktioita etsitään.



Kuva 10. Funktioiden listaus kirjastossa

Kun haluttu funktio on löydetty, voidaan se joko ladata suoraan omalle koneelle tai siirtyä tarkastelemaan funktion manuaaleja. Manuaalisivulla on listattu kaikki vaadituiksi määritellyt funktiot sekä tarkasteltavan funktion revisiohistoria.



Kuva 11. Funktion Manuaalisivu

8 YHTEENVETO

Työn tuloksena saatiin toimiva kirjastosovellus, joka soveltuu useille PLC-alustoille. Käyttökokemuksen myötä kirjastoa voidaan jatkossa kehittää ja täydentää uusilla ominaisuuksilla. Tavoitteisiin nähden tämä osuus siis onnistui hyvin. Työn toisena osuutena ollut ohjelmointioppaan kehitys jäi hieman vähemmälle huomiolle rajoittuen tekovaiheessa vain funktioiden kirjastokäytön aiheuttamiin vaatimuksiin. Tätä osuutta on jatkossa tarkoitus kehittää yhdessä muun ohjelmointiryhmän kanssa.

LÄHTEET

http://en.wikipedia.org/wiki/Relational_database Viitattu 17.2.2012

http://en.wikipedia.org/wiki/MySQL Viitattu 17.1.2012

http://en.wikipedia.org/wiki/Php Viitattu 12.9.2008